

Implementing the AsyncTaskBarrier Framework Using Project Reactor (Part 1)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

- Understand the API of the `AsyncTaskBarrier` class for Project Reactor

Class `AsyncTaskBarrier`

```
public class AsyncTaskBarrier
extends java.lang.Object
```

This class asynchronously runs tasks that use the Project Reactor framework and ensures that the calling method doesn't exit until all asynchronous task processing is completed.

Method Summary

All Methods

Static Methods

Concrete Methods

Modifier and Type	Method	Description
static void	<code>register</code> (<code>java.util.function.Supplier<reactor.core.publisher.Mono<java.lang.Void>></code> task)	Register the task task so that it will be run asynchronously when <code>runTasks()</code> is called.
static <code>reactor.core.publisher.Mono<java.lang.Long></code>	<code>runTasks()</code>	Run all the register tasks.

See [Reactive/flux/ex4/src/main/java/utls/AsyncTaskBarrier.java](https://github.com/reactor/reactor-core/blob/main/src/main/java/utls/AsyncTaskBarrier.java)

Learning Objectives in this Part of the Lesson

- Understand the API of the `AsyncTaskBarrier` class for Project Reactor
- Know how to use `AsyncTaskBarrier` in practice

```
AsyncTaskBarrier.register(this::syncThrowException);  
AsyncTaskBarrier.register(this::asyncThrowException);  
AsyncTaskBarrier.register(this::syncNoException);  
AsyncTaskBarrier.register(this::asyncNoException);
```

```
long testCount = AsyncTaskBarrier  
    .runTasks()  
    .blockingGet();
```

```
assertEquals(testCount, 2);
```

See [Reactive/flux/ex4/src/test/java/Utils/AsyncTaskBarrierTests.java](https://github.com/reactor/reactor-core/blob/main/src/test/java/org/reactor/flux/ex4/src/test/java/Utils/AsyncTaskBarrierTests.java)

The AsyncTask Barrier Class API

The AsyncTaskBarrier Class API

- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks

<<Java Class>>

 **AsyncTaskBarrier**

  sTasks: List<Supplier<Mono<Void>>>

 AsyncTaskBarrier()

 register(Supplier<Mono<Void>>):void

 unregister(Supplier<Mono<Void>>):boolean

 runTasks():Mono<Long>

The AsyncTaskBarrier Class API

- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks
- It provides methods that register & unregister tasks passed as Suppliers

Interface Supplier<T>

Type Parameters:

T - the type of results supplied by this supplier

Functional Interface:

This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

<<Java Class>>

AsyncTaskBarrier

  sTasks: List<Supplier<Mono<Void>>>

 AsyncTaskBarrier()

 register(Supplier<Mono<Void>>):void

 unregister(Supplier<Mono<Void>>):boolean

 runTasks():Mono<Long>

See docs.oracle.com/javase/8/docs/api/java/util/function/Supplier.html

The AsyncTaskBarrier Class API

- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks
- It provides methods that register & unregister tasks passed as Suppliers
 - These tasks are stored in a List

<<Java Class>>	
 AsyncTaskBarrier	
 	sTasks: List<Supplier<Mono<Void>>>
	AsyncTaskBarrier()
	register(Supplier<Mono<Void>>):void
	unregister(Supplier<Mono<Void>>):boolean
	runTasks():Mono<Long>

See docs.oracle.com/javase/8/docs/api/java/util/List.html

The AsyncTaskBarrier Class API

- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks
 - It provides methods that register & unregister tasks passed as Suppliers
 - It also provides a method that runs all registered tasks (a)synchronously

<<Java Class>>	
G AsyncTaskBarrier	
S F	sTasks: List<Supplier<Mono<Void>>>
C	AsyncTaskBarrier()
S	register(Supplier<Mono<Void>>):void
S	unregister(Supplier<Mono<Void>>):boolean
S	runTasks():Mono<Long>

The AsyncTaskBarrier Class API

- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks
 - It provides methods that register & unregister tasks passed as Suppliers
 - It also provides a method that runs all registered tasks (a)synchronously
 - This method doesn't block

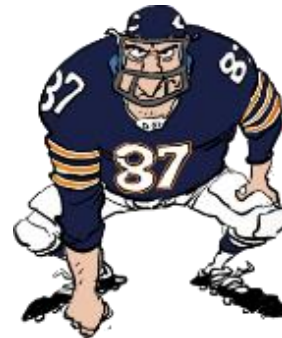
<<Java Class>>	
G AsyncTaskBarrier	
S F	sTasks: List<Supplier<Mono<Void>>>
C	AsyncTaskBarrier()
S	register(Supplier<Mono<Void>>):void
S	unregister(Supplier<Mono<Void>>):boolean
S	runTasks():Mono<Long>



The AsyncTaskBarrier Class API

- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks
 - It provides methods that register & unregister tasks passed as Suppliers
 - It also provides a method that runs all registered tasks (a)synchronously
 - This method doesn't block
 - When combined with Mono.block() the calling thread won't exit until all asynchronous task processing completes

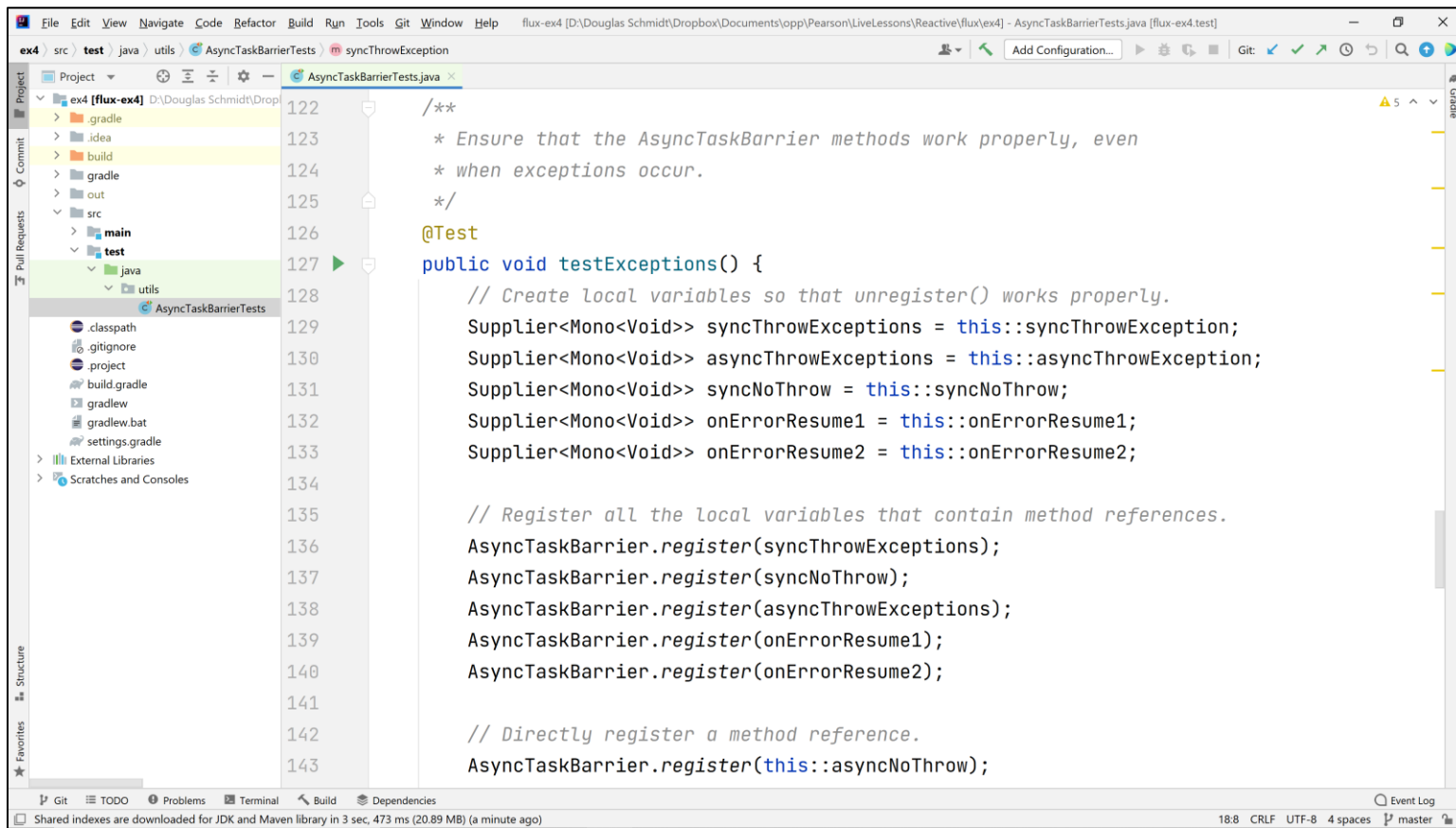
<<Java Class>>	
G AsyncTaskBarrier	
S F	sTasks: List<Supplier<Mono<Void>>>
C	AsyncTaskBarrier()
S	register(Supplier<Mono<Void>>):void
S	unregister(Supplier<Mono<Void>>):boolean
S	runTasks():Mono<Long>



See projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#block

Applying the AsyncTask Barrier in Practice

Applying the AsyncTaskBarrier in Practice



```
122  /**
123   * Ensure that the AsyncTaskBarrier methods work properly, even
124   * when exceptions occur.
125   */
126  @Test
127  public void testExceptions() {
128      // Create local variables so that unregister() works properly.
129      Supplier<Mono<Void>> syncThrowExceptions = this::syncThrowException;
130      Supplier<Mono<Void>> asyncThrowExceptions = this::asyncThrowException;
131      Supplier<Mono<Void>> syncNoThrow = this::syncNoThrow;
132      Supplier<Mono<Void>> onErrorResume1 = this::onErrorResume1;
133      Supplier<Mono<Void>> onErrorResume2 = this::onErrorResume2;
134
135      // Register all the local variables that contain method references.
136      AsyncTaskBarrier.register(syncThrowExceptions);
137      AsyncTaskBarrier.register(syncNoThrow);
138      AsyncTaskBarrier.register(asyncThrowExceptions);
139      AsyncTaskBarrier.register(onErrorResume1);
140      AsyncTaskBarrier.register(onErrorResume2);
141
142      // Directly register a method reference.
143      AsyncTaskBarrier.register(this::asyncNoThrow);
```

See [Reactive/flux/ex4/src/test/java/utis/AsyncTaskBarrierTests.java](https://github.com/reactive/flux-ex4/src/test/java/utis/AsyncTaskBarrierTests.java)

End of Implementing the AsyncBarrierTask Framework Using Project Reactor (Part 1)